Improving the Bundle Creator

About

Contributor : Srirupa Datta
Organization : KDE (Krita)
Project Size : 350 Hours
Project Timeline : Standard

Introduction and Motivation

The primary format to share resources in Krita is a Resource Bundle, which is a compressed file containing all the resources together. It also contains some other information like metadata and a manifest so Krita can check there's no errors in the file. Krita's Bundle Creator allows one to create their own bundle from the resources of their choice. The project aims to improve the user interface of the current Bundle Creator, and allow the ability to edit bundles (which is currently not supported in Krita).

Requirements for the Bundle Creator:

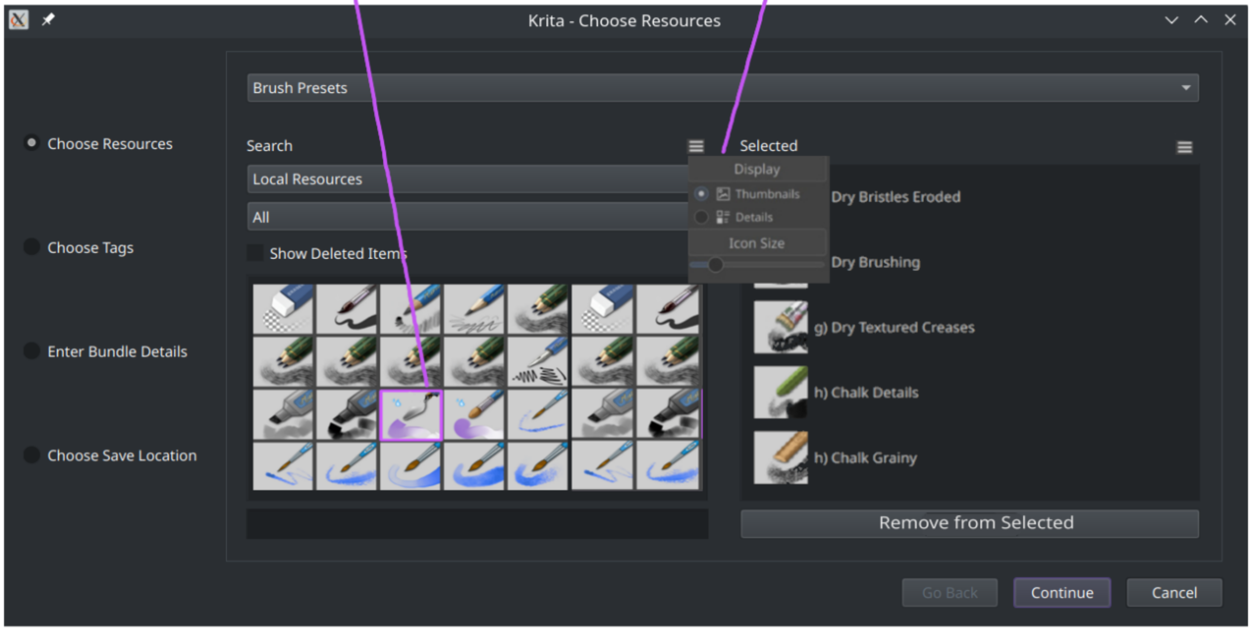
1. Gridview to view all the resources
2. Filter available resources by name or tag before selecting them
3. Add custom tags to selected resources
4. Reload last bundle data when opened - particularly useful when making a bundle for other people.
5. Resizable Bundle Creator - this would be useful for users who want to spend more time creating a new bundle
6. Ability to edit Bundles - this needs to be discussed further
7. Create a separate Menu entry called Bundle Creator. Move Manage Resource Libraries , Manage Resources and Bundle Creator from Menu > Settings to Menu > Resources

Bundle Creator UI

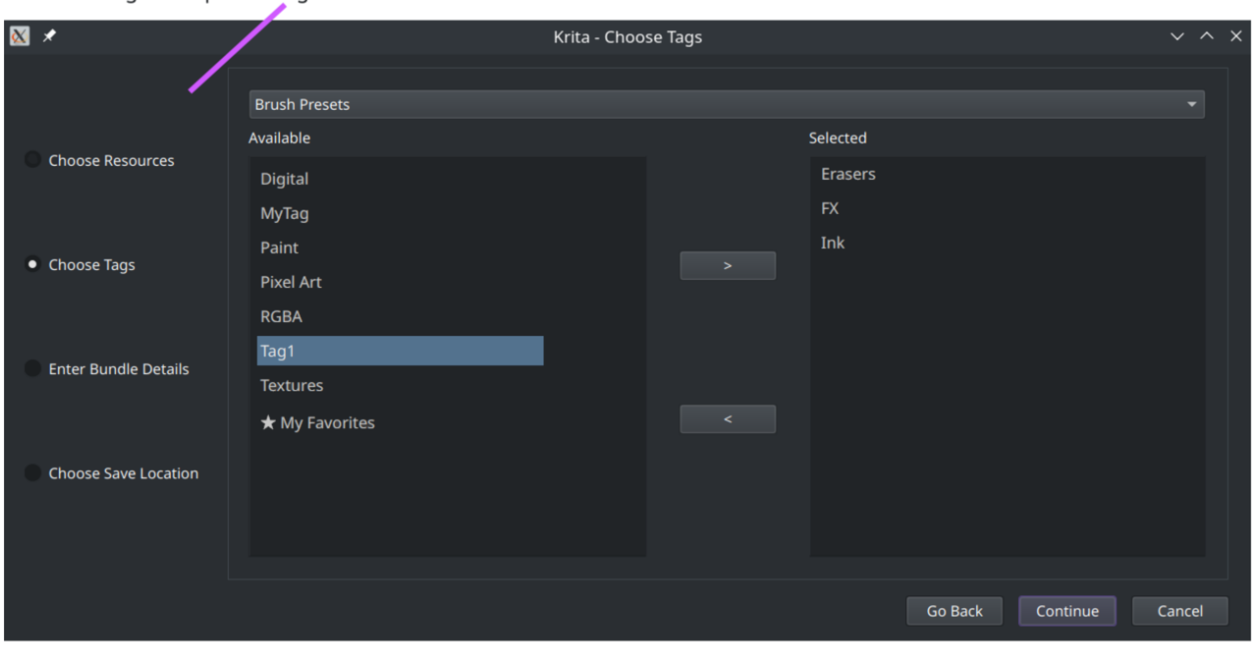
The new Bundle Creator would look like an installation wizard with four pages which can be navigated using the `Next` and `Back` buttons, as well as buttons on the left side panel.

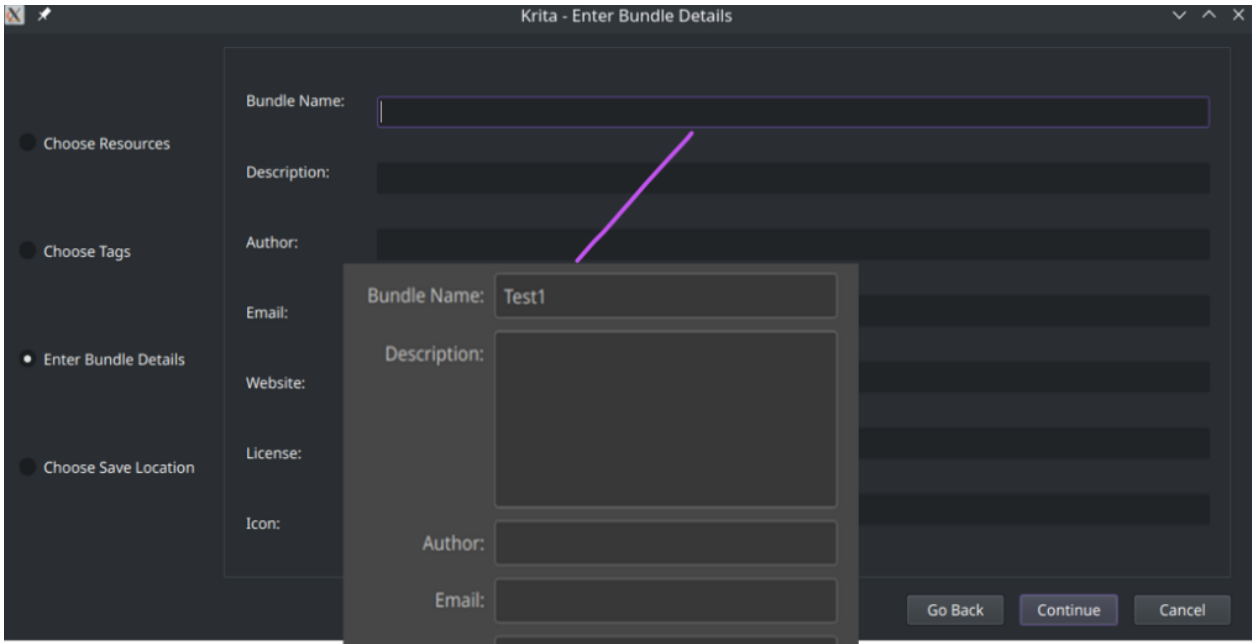
Click to select, click again to deselect resources in the gridview under `Search`. This will add the resources in the adjacent `Selected` listview.

An option to switch from listview to gridview and vice versa for both the `Search` and `Selected` sections. Note the `Thumbnail` and `Details` options under the small display widget



Side Widget for quick navigation between the sections.

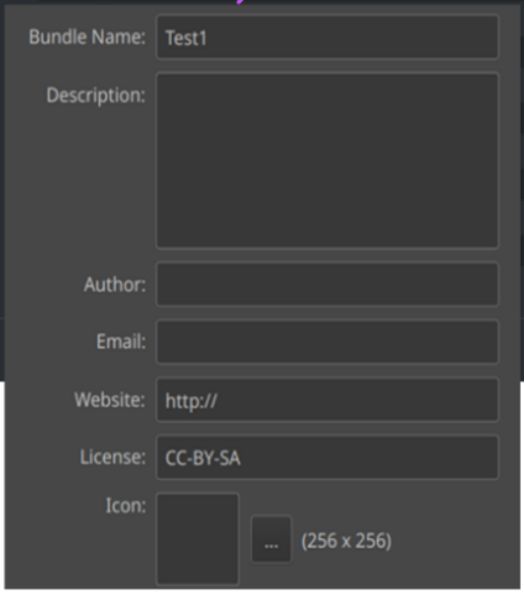



Krita - Enter Bundle Details

- Choose Resources
- Choose Tags
- ● Enter Bundle Details
- Choose Save Location

Bundle Name:	
Description:	
Author:	
Email:	
Website:	
License:	
Icon:	

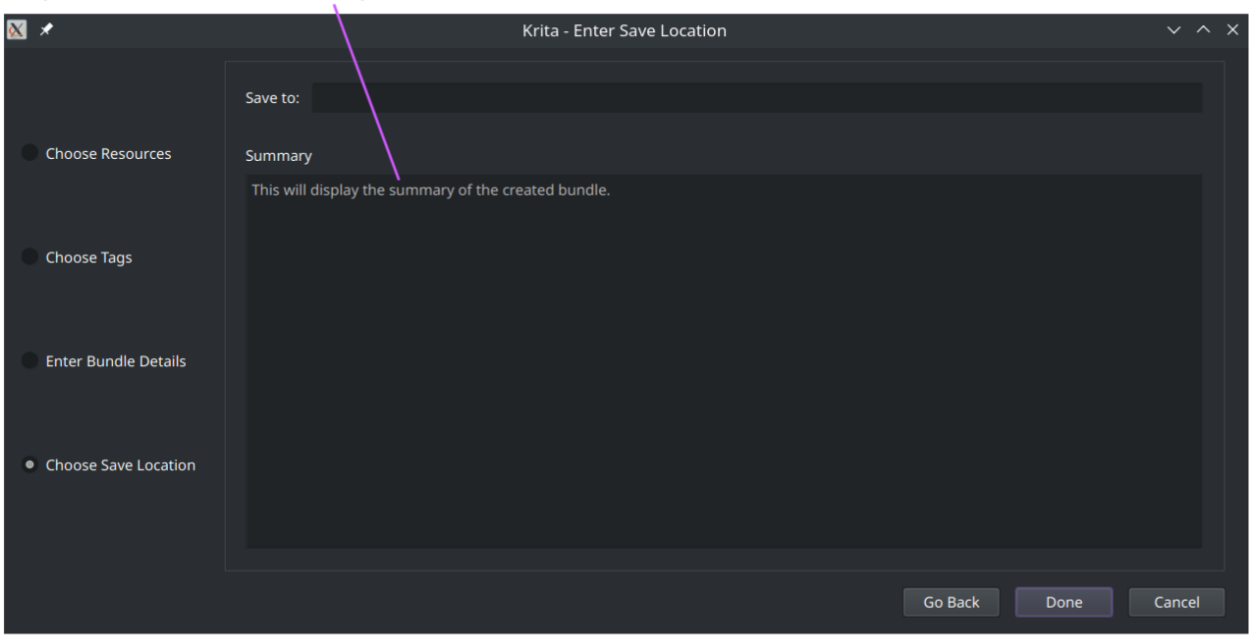
Go Back | Continue | Cancel



Bundle Name:	Test1
Description:	
Author:	
Email:	
Website:	http://
License:	CC-BY-SA
Icon:	... (256 x 256)

This part would look similar to the part shown on the left.

Details of the created bundle would be displayed.
(Details need to be discussed)


Krita - Enter Save Location

- Choose Resources
- Choose Tags
- Enter Bundle Details
- ● Choose Save Location

Save to:

Summary

This will display the summary of the created bundle.

Go Back | Done | Cancel

Project Goals

- Bundle Creator should look like an installation wizard like dialog
- Krita should be able to save and load previous bundle information
- Krita should be able to edit bundles in the Bundle Creator

Implementation

1. Create new UI file and a superclass for common UI

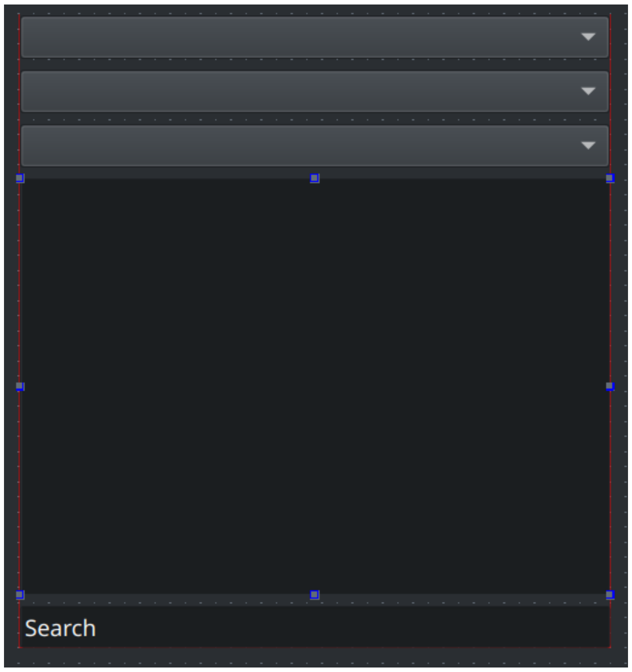
This will contain the UI which is common to both the Resource Manager and the Bundle Creator. The `wdgResourcePreview.ui` file will contain the common template that will be used by both. And `KisResourcePreview.h` will implement the functionalities. It would have the following structure:

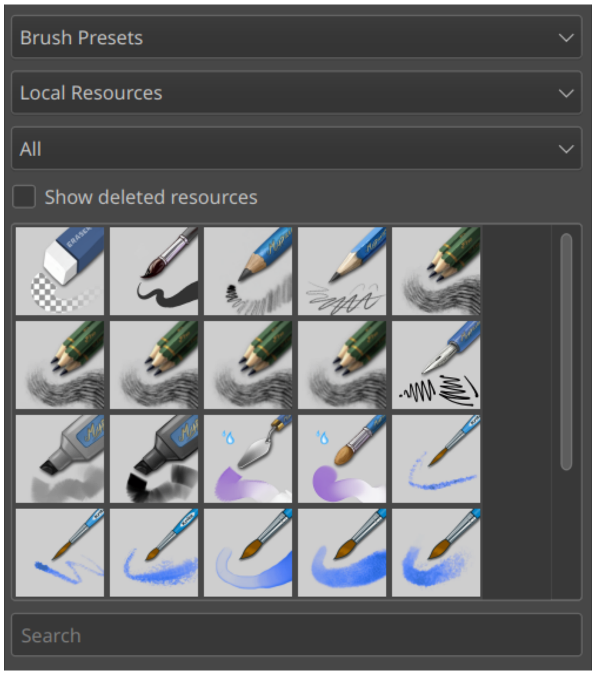
```
namespace Ui
{
class WdgResourcePreview;
}

class ResourcePreview : public QWidget
{
    Q_OBJECT
public:
    ResourcePreview(int type, QWidget *parent = 0);
    ~ResourcePreview();

private Q_SLOTS:
    void slotResourceTypeSelected(int);
    void slotTagSelected(int);
    void slotStorageSelected(int);

    void slotFilterTextChanged(const QString& filterText);
    void slotShowDeletedChanged(int type, int newState);
public:
    QString getCurrentResourceType();
    QModelIndexList geResourceItemsSelected();

private:
    int getCurrentStorageId();

private:
    Ui::WdgResourcePreview *m_ui;
    KisResourceTypeModel *m_resourceTypeModel {0};
    KisStorageModel *m_storageModel {0};

    QMap<QString, KisTagFilterResourceProxyModel*>
m_resourceProxyModelsForResourceType;
    KisResourceThumbnailPainter m_thumbnailPainter;

};
```

The methods and slot functions will be implemented in the `KisResourcePreview.cpp`. To add the common UI to both the Resource Manager and the Bundle Creator, the new widget class has to be added.

```
// For Resource Manager
ResourcePreview *resourcePreview = new ResourcePreview();
ui->splitter->addWidget(resourcePreview);

// For Bundle Creator
ResourcePreview *resourcePreview = new ResourcePreview();
ui->splitter->addWidget(resourcePreview);
```

The `wdgResourcePreview.ui` file would look something like this:





2. Redesigning the Bundle Creator UI

The Bundle Creator would be redesigned from a normal `Qdialog` to a `Qwizard` with 4 pages. This is what the new Bundle Creator wizard class would look like:

```
namespace Ui {
    class WzdCreateBundle;
}

class WzdCreateBundle : public QWizard, public KoDialog
{
    Q_OBJECT

public:
    WzdCreateBundle(QWidget *parent = 0);
    ~WzdCreateBundle();

public Q_SLOTS:
    void openSelectResources();
    void openSelectTags();
    void openMetadataInfo();
    void openSelectSaveLocation();
private:
    Ui::WzdCreateBundle *m_ui;

    PageCreateBundleResourceChooser *pageSelectResource;
    PageCreateBundleTagChooser *pageSelectTags;
    PageCreateBundleMetadataInfo *pageMetadataInfo;
    PageCreateBundleSaver *pageSelectSaveLocation;
};
```

The Bundle Creator wizard will have four pages added to it. We will create four classes that inherit `QWizardPages` to add these pages to the wizard. The left side of the wizard will have four buttons for easy navigation to the pages. This will be done by setting the side widget for the Bundle Creator using:

```
setSideWidget(sideWidget);
```

This is how the side widget class would look like:

```
namespace Ui {
    class SideWdg;
}

class SideWdg : public QWidget
{
    Q_OBJECT

public:
    explicit SideWdg(QWidget *parent = 0);
    ~SideWdg();

signals:
    void openSelectResourcesClicked();
    void openSelectTagsClicked();
    void openMetadataInfoClicked();
    void openSelectSaveLocationClicked();

private:
    Ui::SideWdg *ui;
};
```

The four `QWizardPage` classes have been described below:

1. KisPageCreateBundleResourceChooser.cpp/h with PageCreateBundleResourceChooser.ui file

This class will deal with the selection of resources for the new bundle.

```
namespace Ui
{
class PageCreateBundleResourceChooser;
}

class PageCreateBundleResourceChooser : public QWizardPage
{
    Q_OBJECT
public:
    PageCreateBundleResourceChooser(QWidget *parent = 0);
    ~PageCreateBundleResourceChooser();

    QList<int> getSelectedResources();

public Q_SLOTS:
    void slotResourcesSelectionChanged(QModelIndex selected);

private Q_SLOTS:
    void slotSelectedResourcesSelectionChanged(QModelIndex selected);
    void slotRemoveSelectedResources();
    void slotNext();

private:
    Ui::PageCreateBundleResourceChooser *m_ui;
    QList<int> m_selectedResourcesIds;
    QMap<QString, KisTagFilterResourceProxyModel*>
m_resourceProxyModelsForResourceType;

};
```

2. KisPageCreateBundleTagChooser.cpp/h with PageCreateBundleTagChooser.ui file

This class will deal with the selection of tags for the new bundle.

```
namespace Ui
{
class PageCreateBundleTagChooser;
}

class PageCreateBundleTagChooser : public QWizardPage
{
    Q_OBJECT
public:
    PageCreateBundleTagChooser(QWidget *parent = 0);
    ~PageCreateBundleTagChooser();

    QList<int> getSelectedTags();

private Q_SLOTS:
    void addSelected();
    void removeSelected();
    void resourceTypeSelected(int idx);
    void slotNext();
    void slotBack();

private:
    Ui::PageCreateBundleTagChooser *m_ui;

    QList<int> m_selectedTagIds;

};
```

3. KisPageCreateBundleMetadataInfo.cpp/h with PageCreateBundleMetadataInfo.ui file

This class will deal with entering the metadata information for the new bundle.

```
namespace Ui
{
class PageCreateBundleMetadataInfo;
}

class PageCreateBundleMetadataInfo : public QWizardPage
{
    Q_OBJECT
public:
    PageCreateBundleMetadataInfo(QWidget *parent = 0);
    ~PageCreateBundleMetadataInfo();

    QString bundleName() const;
    QString authorName() const;
    QString email() const;
    QString website() const;
    QString license() const;
    QString description() const;
    QString previewImage() const;

private Q_SLOTS:
    void getPreviewImage();
    void slotNext();
    void slotBack();

private:
    Ui::PageCreateBundleMetadataInfo *m_ui;

    QString m_previewImage;
};
```

4. KisPageCreateBundleSaver.cpp/h with PageCreateBundleSaver.ui file

This class will deal with entering the location where the new bundle will get saved.

```
namespace Ui
{
class PageCreateBundleSaver;
}

class PageCreateBundleSaver : public QWizardPage
{
    Q_OBJECT
public:
    PageCreateBundleSaver(QWidget *parent = 0);
    ~PageCreateBundleSaver();

    QString saveLocation() const;

private Q_SLOTS:
    void accept() override;

    void selectSaveLocation();
    void saveToConfiguration(bool full);
    void slotBack();

private:
    bool putResourcesInTheBundle(KoResourceBundleSP bundle);
    void putMetaDataInTheBundle(KoResourceBundleSP bundle) const;
    QString createPrettyFilenameFromName(KoResourceSP resource) const;

    Ui::PageCreateBundleSaver *m_ui;

    KoResourceBundleSP m_bundle;

};
```

2. Save and Load previous info in the Bundle Creator

Create functions in the `WzdCreateBundle.h/cpp` to load previous bundle information on opening the Bundle Creator.

```
WzdCreateBundle::void loadUISettings(const QString &prefix = QString());
WzdCreateBundle::void saveUISettings(const QString &prefix = QString());
```

3. Editing Bundles inside the Bundle Creator

Create a dialog with all the existing bundles listed. Choosing a menu entry and clicking a button, opens up the bundle creator filled with all information loaded from the bundle(which was selected). Then on pressing `OK` or `Save`, the bundle's information gets saved. Depending on where one saves the bundle, and whether the metadata info is changed, the existing bundle is rewritten or a new bundle is created.

Benefits

- Users can filter the resources to be selected by tags or by name.
- Existing bundles can be edited.
- Bundle Creator will retain previous bundle information. This is particularly useful when making a bundle for other people.
- A resizable Bundle Creator Window will be useful for users who want to spend more time creating a new bundle. Looking into a medium sized window is not always convenient.

Timeline

Week	Timeline	Task
	May 4	Accepted GSoC contributor projects announced
	May 4 - May 20	Community bonding period - Learn about the MVC and MVVM design patterns - Explore Krita’s Resource Management System for understanding the cause of resource duplication
	May 29	Coding Period Begins
1	May 29 - Jun 4	- Design KisResourcePreview.h properly with all the relevant signals, slots and methods (as mentioned above) - This header file will contain classes responsible for the common UI between the Resource Manager and the Bundle Creator.
2	Jun 5 - Jun 11	- Implement the .cpp file for the common UI - Reflect the relevant changes in the DlgResourceManager.h/cpp (Resource Manager Dialog)
3	Jun 12 - Jun 18	- Design header files for the new Bundle Creator wizard - Discuss about redesigning the UI for the tag chooser
4	Jun 19 - Jun 25	- Implement the .cpp files for the Resource Chooser - Implement the .cpp files for the Tag Chooser - Implement the ability to add custom tags to selected resource items
5	Jun 26 - Jul 2	
6	Jul 3 - Jul 9	- Implement the .cpp files for the Bundle Information Dialog - Discuss what to summarize in the Save to Dialog - Implement the .cpp files for the Save to Dialog
	Jul 10	Mentors and Contributors start submitting midterm evaluations
	Jul 14	Midterm evaluation deadline
7	Jul 17 - Jul 23	- Add support for saving previous bundle Information - Discuss details about adding support for editing bundles
8	Jul 24 - Jul 30	- Design the user interface for the Bundle Editor - Implement the functions for the Bundle Editor - Integrate it with the Bundle Creator
9	Jul 31 - Aug 6	
10	Aug 7 - Aug 13	
11	Aug 14 - Aug 20	- User testing - Rewriting the documentation for the Bundle Creator
12	Aug 21 - Aug 28	- Clean up work, complete unfinished tasks
	Aug 28	End of Google Summer of Code

About Me

Name: Srirupa Datta
IRC: Srirupa Datta (@sriru:libera.chat)
KDE identity: srirupa
Email: srirupa.sps@gmail.com
Krita Artists: Srirupa_Datta
GitHub: Github
LinkedIn: srirupa-datta
University: Jadavpur University
Country: India
Primary language: English, Bengali, Hindi

Communication

I plan to communicate in the IRC channel on a regular basis (maybe once every 2-3 days) to update about my progress and clarify my doubts. During the weekly meetings at Krita, I would summarize the work I did during the week, and the challenges I am facing. I will also post a more formal blog update to PlanetKDE once every month and continue updating my progress on the Krita-artists forum on the post that I've already created. Link: Improving the Bundle Creator

Prior Contributions

Being an artist who is interested in coding, I was excited to contribute to Krita, and went on to learn Qt. Since then I have been fixing minor bugs in Krita whenever I get time. The issues that I've fixed are listed below:

- Pull Request !801
  Added a warning dialog whenever a file with multiple transparency masks is exported to a file format that does not support multiple transparency masks (such as Photoshop) preventing silent loss of data.
- Pull Request !894
  Fixed a bug that made Krita freeze when splitting layers on a colorize mask by automatically converting the colorize mask to a paint layer which is faster to split.

- Pull Request !883
  Fixed a bug that showed the wrong preview for brushes that did not have a preview available by modifying the order of drawing things on the preview view.
- Pull Request !854
  Removed the work recursively button which was unnecessary and ambiguous and cleaned up the code for that feature, reducing the code complexity in transform tool
- Pull Request !795
  Reordered the touch docker to make the display more intuitive for users.
- Pull Request !886
  Swapped the y axis labels in the transfer curve of brush mix parameter

I have also contributed to Krita as a part of Season of KDE 2022, where I worked on adding the Perspective Ellipse Assistant Tool to Krita. The merge request can be found here.